IT-314 Software Engineering

Lab - 09 – Mutation Testing

Name : Shyam Ghetiya

Student-Id : 202201161

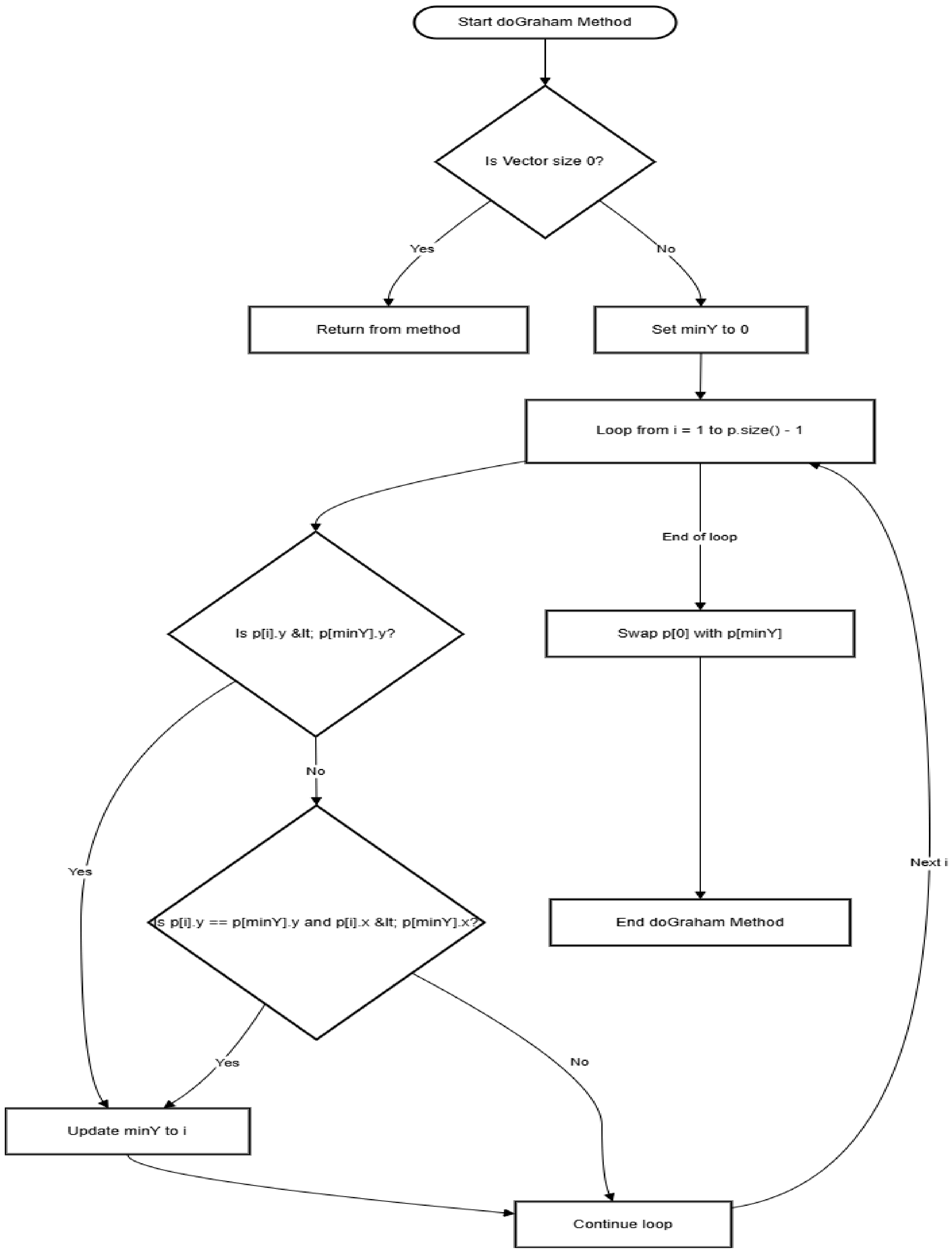
Aim: – Mutation Testing

Q.1. **The code below is part of a method in the ConvexHull class in the VMAP system. The following is a small fragment of a method in the ConvexHull class. For the purposes of this exercise, you do not need to know the intended function of the method. The parameter p is a Vector of Point objects, p.size() is the size of the vector p, (p.get(i)).x is the x component of the ith point appearing in p, similarly for (p.get(i)).y. This exercise is concerned with structural testing of code, so the focus is on creating test sets that satisfy some particular coverage criteria.**

Answer :

```java
public class Point {
    double x;
    double y;
    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}
public class ConvexHull {
    public void doGraham(Vector<Point> p) {
        if (p.size() == 0) {
            return;
        }
        int minY = 0;
        for (int i = 1; i < p.size(); i++) {
            if (p.get(i).y < p.get(minY).y ||
            (p.get(i).y == p.get(minY).y && p.get(i).x <
            p.get(minY).x)) {
                minY = i;
            }
        }
        Point temp = p.get(0);
        p.set(0, p.get(minY));
        p.set(minY, temp);
    }
}
```

- **This is the Control Flow Graph of above code :**



Q.2. **Construct test sets for your flow graph that are adequate for the following criteria:**

a. Statement Coverage.

b. Branch Coverage.

c. Basic Condition Coverage.

Answer :

1. Statement Coverage :

Test Cases:

- **Test Case 1 :** Empty vector p (i.e., p.size() == 0).
  - Expected Outcome: The method should exit immediately without further processing.
- **Test Case 2 :** Vector p with a single Point (e.g., p = [Point(1, 1)]).
  - Expected Outcome: The method should initialize minY to 0 and skip the loop, as there's only one point.
- **Test Case 3 :** Vector p with multiple points, all having the same y-coordinate.
  - Example Input: p = [Point(2, 2), Point(1, 2), Point(3, 2)].
  - Expected Outcome: The method should select the point with the smallest x-coordinate as minY, and after the loop, it swaps p[0] with p[minY].

2. Branch Coverage :

Test Cases:

- **Test Case 1 :** Provide an empty vector p (i.e., p.size() == 0).
    - Expected Outcome: The condition if (p.size() == 0) evaluates to true, and the method exits.
- **Test Case 2 :** Vector p with one point (e.g., p = [Point(0, 0)]).
    - Expected Outcome: The condition if (p.size() == 0) evaluates to false, and the method skips the loop due to only one point.
- **Test Case 3 :** Vector p with two points with different y-coordinates, where p[1].y < p[0].y.
    - Example Input: p = [Point(1, 3), Point(1, 1)].
    - Expected Outcome: The condition p[i].y < p[minY].y is true for i = 1, updating minY to 1. After the loop, p[0] swaps with p[minY].
- **Test Case 4 :** Vector p with multiple points where both p[i].y == p[minY].y and p[i].x < p[minY].x hold true at least once.
    - Example Input: p = [Point(2, 2), Point(1, 2), Point(3, 2)].
    - Expected Outcome: Both conditions hold for i = 1, so minY is updated to 1, and p[0] swaps with p[minY].
- **Test Case 5 :** Vector p with points where no point has a smaller y-coordinate or smaller x-coordinate than p[0].
    - Example Input: p = [Point(1, 1), Point(2, 2), Point(3, 3)].

    - Expected Outcome: The loop completes without any updates to minY, leaving p[0] unchanged.

3. Condition Coverage :

Test Cases:

- **Test Case 1** : Vector p with a single point.
    - Example Input: p = [Point(0, 0)].
    - Expected Outcome: The condition if (p.size() == 0) is false, allowing the method to initialize minY and skip the loop as there’s only one point.
- **Test Case 2** : Vector p with two points where p[i].y < p[minY].y is true.
    - Example Input: p = [Point(3, 3), Point(1, 2)].
    - Expected Outcome: The condition p[i].y < p[minY].y evaluates to true, updating minY to 1. The loop completes, and p[0] swaps with p[minY].
- **Test Case 3** : Vector p with multiple points where only p[i].x < p[minY].x is true (i.e., points with the same y-coordinate, but different x-coordinates).
    - Example Input: p = [Point(2, 1), Point(1, 1), Point(3, 1)].
    - Expected Outcome: The condition p[i].y == p[minY].y is true, and p[i].x < p[minY].x is true for i = 1, updating minY to 1. After the loop, p[0] swaps with p[minY].

- **Test Case 4 :** Vector p where neither p[i].y < p[minY].y nor p[i].x < p[minY].x conditions are true.
    - Example Input: p = [Point(1, 1), Point(3, 3), Point(5, 5)].
    - Expected Outcome: No conditions are met, so minY remains 0 and p[0] remains unchanged.
- **Test Case 5 :** Vector p where p[i].y == p[minY].y but p[i].x < p[minY].x is false.
    - Example Input: p = [Point(0, 1), Point(1, 1), Point(2, 1)].
    - Expected Outcome: Since p[i].x < p[minY].x is never true, minY remains 0, and no swap occurs.

Q.3. For the test set you have just checked can you find a mutation of the code (i.e. the deletion, change or insertion of some code) that will result in failure but is not detected by your test set. You have to use the mutation testing tool.

Answer :

1. **Deletion Mutation :**

    Mutation: Remove the initialization of minY to 0 at the start of the method.

```java
public class ConvexHull {

    public void doGraham(Vector<Point> p) {

        if (p.size() == 0) {

            return;

        }

        // int minY = 0;  // Initialization removed

        for (int i = 1; i < p.size(); i++) {

            if (p.get(i).y < p.get(minY).y ||

                (p.get(i).y == p.get(minY).y && p.get(i).x <
p.get(minY).x)) {

                minY = i;

            }

        }

        Point temp = p.get(0);

        p.set(0, p.get(minY));

        p.set(minY, temp);

    }

}
```

Impact: Omitting the initialization of minY causes it to be unassigned when accessed in the loop. This can lead to unpredictable behavior or even a runtime error if minY is used without being properly set.

2. Insertion Mutation :

Mutation: Add a line that conditionally overrides `minY` to an incorrect value.

```
public class ConvexHull {
    public void doGraham(Vector<Point> p) {
        if (p.size() == 0) {
            return;
        }
        int minY = 0;
        if (p.size() > 1) {
            minY = 1;  // Incorrectly setting minY to 1
        }
        for (int i = 1; i < p.size(); i++) {
            if (p.get(i).y < p.get(minY).y ||
                (p.get(i).y == p.get(minY).y && p.get(i).x <
p.get(minY).x)) {
                minY = i;
            }
        }
        Point temp = p.get(0);
        p.set(0, p.get(minY));
        p.set(minY, temp);
    }
}
```

Impact: This mutation sets minY to 1 if there is more than one point, potentially overriding the actual minimum y-coordinate selection logic.

3. Modification Mutation :

Mutation: Change the logical OR (||) to an AND (&&) in the conditional statement inside the loop.

```
public class ConvexHull {
    public void doGraham(Vector<Point> p) {
        if (p.size() == 0) {
            return;
        }
        int minY = 0;
        for (int i = 1; i < p.size(); i++) {
            if (p.get(i).y < p.get(minY).y &&
                (p.get(i).y == p.get(minY).y && p.get(i).x <
p.get(minY).x)) {
                minY = i;
            }
        }
        Point temp = p.get(0);
        p.set(0, p.get(minY));
        p.set(minY, temp);
    }
}
```

Impact: Replacing the || operator with && in the condition changes the logic for finding the minimum y-coordinate. Now, minY is only updated if both the y-coordinates are equal and the x-coordinate is smaller, which is not the intended behavior.

Analysis of Detection by Test Set :

1. **Statement Coverage**:
   - The deletion of `minY` initialization might not be caught by statement coverage alone, as it could lead to undefined behavior depending on the input but may not trigger an explicit error if no point accesses `minY` directly.

2. **Branch Coverage**:
   - The mutation that assigns `minY` to 1 can result in incorrect outcomes if the code does not correctly identify the minimum y-coordinate point. However, if the tests don't confirm the ordering or positioning of points in the result, this mutation might go unnoticed.

3. **Condition Coverage**:
   - Changing `||` to && in the conditional expression doesn't result in an immediate crash, but it modifies the selection logic for `minY`. Without checking if `minY` is correctly updated under varied y and x values, this mutation may escape detection in condition-based test cases.

Q4. Create a test set that satisfies the path coverage criterion where every loop is explored at least zero, one or two times :

Test Case 1: Loop Not Entered (Zero Iterations)

- **Input**: An empty vector `p`.
- **Setup**: `Vector<Point> p = new Vector<Point>();`
- **Expected Outcome**: The method should terminate immediately without any processing since `p` has no elements.

This verifies the behavior when `p.size()` is zero, triggering the early exit condition.

Test Case 2: Loop with a Single Element (One Iteration)

- **Input** : A vector containing one point.
- **Setup :**

```
Vector<Point> p = new Vector<Point>();
p.add(new Point(0, 0));
```

- **Expected Outcome:** Since `p.size()` is 1, the loop should not execute, as there are no other elements to compare. The point swaps with itself, leaving p unchanged. This checks the scenario where the loop effectively runs once.

Test Case 3: Loop Executed Twice (Two Iterations)

- **Input**: A vector with two points, where the first point has a higher y-coordinate than the second.
- **Setup :**

```
Vector<Point> p = new Vector<Point>();

p.add(new Point(1, 1));

p.add(new Point(0, 0));
```

- **Expected Outcome:** The loop compares both points and identifies the second point as having a lower y-coordinate. As a result, `minY` is updated to 1, and a swap occurs, placing the second point at the start of the vector.

Test Case 4: Loop Executed Multiple Times

- **Input**: A vector with multiple points.

Setup:

```
Vector<Point> p = new Vector<Point>();

p.add(new Point(2, 2));

p.add(new Point(1, 0));

p.add(new Point(0, 3));
```

- **Expected Outcome**: The loop iterates over all three points. The point `(1, 0)` has the lowest y-coordinate, so `minY` is updated to 1. A swap places `(1, 0)` at the start of the vector, ensuring correct ordering based on y-coordinates.

Q.1. After generating the control flow graph, check whether your CFG matches with the CFG generated by Control Flow Graph Factory Tool and Eclipse flow graph generator.

Answer :

- Control Flow Graph Factory :- YES
- Eclipse flow graph generator :- YES

Q.2. Devise the minimum number of test cases required to cover the code using the aforementioned criteria.

Answer :

Statement Coverage: 3 test cases

1. Branch Coverage: 4 test cases
2. Basic Condition Coverage: 4 test cases
3. Path Coverage: 3 test cases

Summary of Minimum Test Cases:

Total : 3 (Statement) + 4 (Branch) + 4 (Basic Condition) + 3 (Path) = 14 test cases

Q.3. and Q.4. Same as Part I